(1) CPU processing

Firstly, based on RTT technique, the lattice data are translated into 2D texture slices. The first texture is set to be the data source of the first iterative calculation.

Each texel of texture corresponds to a cloud particle. The spherical harmonic coefficients and density are stored in RGBA channels. As shown in Fig. 7.

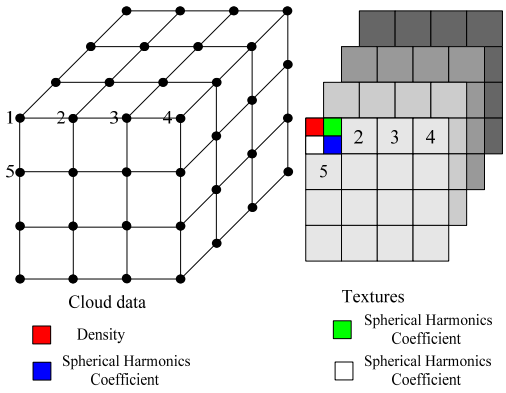


Fig. 7 Cloud data storage structure

In the following calculation process, the spherical harmonic coefficients are updated according to the iterative calculation results.

(2) GPU processing

Firstly, using equation (19), spherical harmonic coefficients of current texture are calculated according to the information of the first texture. The results of the first iteration are set to be the input data for the following iterative calculation. This procedure can be repeated, until all the textures are processed.

6 Cloud Rendering based on Frequency Domain Volume Rendering

The most popular rendering techniques for 3D cloud simulation include splatting [6, 7] and slice-based volume rendering [8, 9, 17]. These methods take full advantage of parallel capability of GPU and achieve high rendering speed. However, when viewpoint changes, the volume clouds need to be reconstructed, the computation cost can not be ignored.

Considering the ever-changing viewpoint, integral calculation and massive cloud data in cloud scene, frequency domain volume rendering [21, 22] is a feasible approach. Nevertheless, traditional frequency domain volume rendering algorithm lacks of depth information. Therefore, we present a frequency domain volume rendering method combined with spherical harmonics to look for the right compromise between realism and efficiency.

6.1 Frequency Domain Volume Rendering Algorithm

The traditional frequency domain volume rendering algorithm is based on Fourier projection slice theorem. Once a volume data is Fourier transformed, an image for any viewing direction can be obtained by extracting a 2D slice of the 3D spectrum at the appropriate orientation and then inverse Fourier transforming it [21, 22]. As shown in Fig. 8.

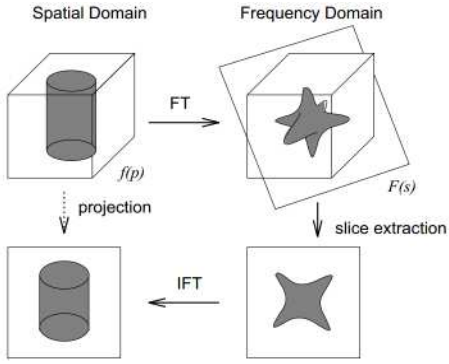


Fig. 8 Frequency-domain volume rendering

The implementation of frequency domain volume rendering includes two steps:

(1) Pre-processing step. 3D spatial data are transformed into the frequency domain, which can be expressed as:

$$F(u,v,w)=\iiint f(x,y,z)e^{-2\pi i(xu+yv+zw)}dxdydz \quad (20)$$

(2) Real-time rendering step. A parallel projection of $f(x,y,z)$ can be computed by evaluating $F(u,v,w)$ along a plane defined by the arbitrary orthonormal vectors. Each point of the plane corresponds to a function value:

$$P_\theta(\omega_u,\omega_v)=F(\omega_u\vec{u}+\omega_v\vec{v}) \quad (21)$$

At last, taking its inverse 2D Fourier transform yields the projection.

The primary motivation of frequency domain volume rendering is that, once the forward 3D transform is computed as a pre-processing operation, we can compute projections at arbitrary angles quickly by working with 2D manifolds of the data in frequency domain. Its essence is linear integral along some directions.

6.2 Frequency Domain Volume Rendering Algorithm Combined with Spherical Harmonics

In real-time rendering phase, equation (11) is used to obtain the light intensity in direction of sight line. As shown in equation (11).

Set $A = T(s, D_n)K(\vec{x}(s))\int_{\Omega} L_{in}(\vec{x}(s), \vec{\omega})P(\vec{\omega}, \vec{\omega}_V)d\vec{\omega}$

Equation (11) can be expressed as:

$$L_{out}(u, v, \vec{\omega}_V) = L_{back}(0, \vec{\omega}_V)T(\vec{x}_{u,v}, D) + \int_s Ads \quad (22)$$

Obviously, $L_{in}(\vec{x}(s), \vec{\omega})$ is represented by a series of spherical harmonic coefficients $c^m_{l(L_{in})}$. $T(\vec{x}_{u,v}, D)$ is optical depth, which can be calcuated as follows:

$$T(\vec{x}, D(\vec{\omega})) = e^{-\int_s^{s'} \sigma\eta(x+t\vec{\omega})dt} \quad (23)$$

η is density. σ depicts albedo. All of them are set to be constant values. Thus, we only need to calculate:

$$F(u, v, w) = \iiint \sigma\eta e^{-2\pi i(xu+yv+zw)}dxdydz \quad (24)$$

(1) Depth information support

Traditional frequency domain volume rendering lacks depth information. Totsuka et al. [22] provided an approach to overcome this drawback. However, it is time consuming. In our method, we adopt a function $\psi(\eta(s))$ related to density to approximately substitute $T(s, D)$. Thus, in equation (22), $\int_s Ads$ can be depicted as:

$$\int_s \psi(\eta(s))K(\vec{x}(s))\int_{\Omega} L_{in}(\vec{x}(s), \vec{\omega})P(\vec{\omega}, \vec{\omega}_V)d\vec{\omega}ds \quad (25)$$

(2) Phase function support

Firstly, we transform equation (25) into the following form:

$$\int_s \psi(\eta(s))K(\vec{x}(s))[\vec{V}(c^m_{l(P)}) \cdot \vec{V}(c^m_{l(L_{in})}(\vec{x}(s)))]ds \quad (26)$$

where $\vec{V}(c^m_{l(L_{in})}(\vec{x}(s)))$ is the spherical harmonic coefficient vector corresponding to incident light distribution. $\vec{V}(c^m_{l(P)})$ is the spherical harmonic coefficient vector corresponding to phase function. Equation (26) can be formed as:

$$\sum_s [\psi(\eta(s))K(\vec{x}(s)) \times \vec{V}(c^m_{l(L_{in})}(\vec{x}(s)))] \cdot \vec{V}(c^m_{l(P)}) \quad (27)$$

The essential of equation (27) is to calculate emergent light intensity accumulation of all the sampling points in direction of sight line. Thus, the phase function of each sampling point is the same. Each sampling point in the integral path has the same $\vec{V}(c^m_{l(P)})$. Therefore,

$$\vec{V}(c^m_{l(P)}(s_1)) = \vec{V}(c^m_{l(P)}(s_2)) = ... = \vec{V}(c^m_{l(P)}(s_n)) \quad (28)$$

Equation (27) can be depicted as:

$$\begin{aligned} &\sum_l \sum_m c^m_{l(P)} \sum_s c^m_{l(L_{in})}(\vec{x}(s))\alpha(\vec{x}(s)) \\ &= V(c^m_{l(P)}) \cdot \left[V(\sum_s c^m_{l(L_{in})}(\vec{x}(s)) \times \alpha(\vec{x}(s))) \right] \\ &= V(c^m_{l(P)}) \cdot \left[V(\int_s c^m_{l(L_{in})}(\vec{x}(s))\alpha(\vec{x}(s))ds) \right] \end{aligned} \quad (29)$$

where $\alpha(\vec{x}(s)) = \psi(\eta(s))K(\vec{x}(s))$.

From a physical standpoint, the essential of our method is to make the incident light distribution of each particle in the integral path equivalent to the total incident light distribution of a single particle. As shown in Fig. 9.

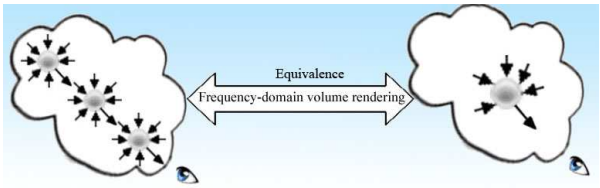
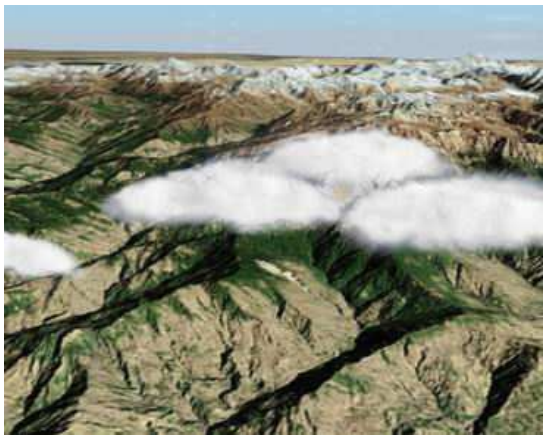


Fig. 9 Physical meaning of phase function-supported frequency domain volume rendering

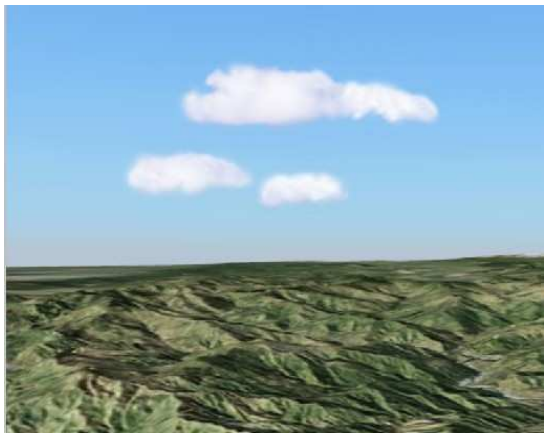
7 Experiments and Discussions

We use OpenGL on C++ platform to implement the method presented in this paper, and carry out related experiments on PC. The computer configuration used for experiments is Windows XP operating system, Intel Pentium 2.6GHz CPU, 2G memory, and NIVDIA 9800GT graphics card.

The performances of our rendering method are shown in Fig.10, which have little perceivable loss in image quality.



(a)

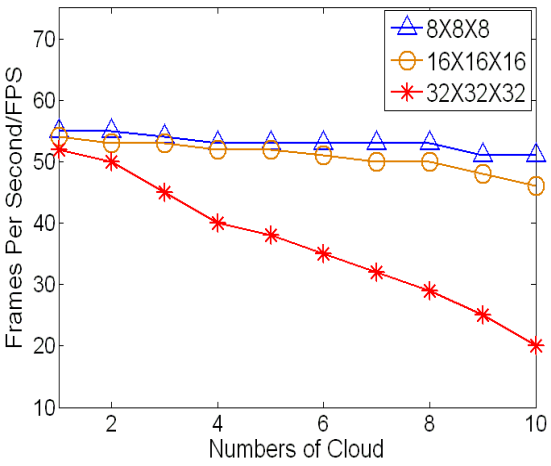


(b)

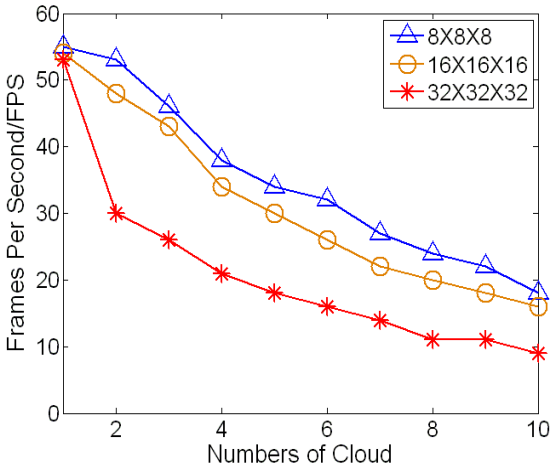
Fig. 10 The snapshots of cloud simulation

To testify the influences of lattice number and interpolation algorithm on the rendering efficiency, we carried out a series of pressure tests. Firstly, we constructed three scenes. The lattice number in above-mentioned three scenes are set to $8\times8\times8$, $16\times16\times16$ and $32\times32\times32$ respectively. Based on tri-linear interpolation and high order interpolation, we analyze rendering efficiency. The fly-through path is predefined. We sampled 2000 frames in each scene respectively, and achieved the average frame rate. Fig.11 shows the result.

For tri-linear interpolation, when the number of lattice is $8\times8\times8$ and $16\times16\times16$, with the increase of cloud number, the rendering efficiency decreased not evidently. When the cloud number is 10, the rendering speed is 50fps and 45fps, respectively. With the increase of lattice point, the rendering efficiency decreased. Benefit from frequency domain volume rendering, the computational cost can be controlled effectively. When the number of lattice is $32\times32\times32$ and the number of rendering cloud is 10, the rendering speed is still 20fps.



(a) Rendering efficiency with tri-linear interpolation



(b) Rendering efficiency with high order interpolation

Fig. 11 Rendering efficiency with different cloud resolution and interpolation algorithm

For high order interpolation, with the increase of lattice point, the rendering efficiency decreased quickly. When the number of lattice is $32\times32\times32$ and the number of rendering cloud is 4, the rendering speed is still 20fps.

8 Conclusion

Realistic simulation of cloud is a challenging problem in the field of natural phenomena simulation with wide applications. In this paper, we have proposed an effective method for simulating 3D cloud. The interactions between light and cloud are considered, and achieved by using a series of spherical harmonics and spherical harmonic coefficients to represent incident-light distribution. To improve rendering speed, a frequency domain volume-rendering algorithm combined with spherical harmonics is presented.

Obviously, the method proposed in this paper is not considering flying through or inside the clouds. Furthermore, our method is not very suitable for animation of clouds. In the near future, we would like to extend our algorithm to handle the dynamic simulation of cloud.

Acknowledgement:

The authors wish to thank the anonymous reviewers for their thorough review and highly appreciate the comments and suggestions, which significantly contributed to improving the quality of this paper.

This work is supported by National High Technology Research and Development Program of China (No. 2012AA011503), Project on the Integration of Industry, Education and Research of

Guangdong Province (No. 2012B090600008) and the pre-research project (No. 51306050102).

References:

[1] H. Qiu, K. Yang, Y. Chen, Survey on realistic simulation of cloud, *Journal of Computer Science*, Vol.38, No.6, 2011, pp. 14-19. (in Chinese)

[2] J. Schopik, J. Simons, D. S. Ebert, C. Hansen, A real-time cloud modeling, rendering, and animations system, *In Proc. Eurographics'03*, Leuven, Belgium, 2003, pp. 160-166.

[3] T. Nishita, Y. Dbashi, E. Nakamae, Display of clouds taking into account multiple anisotropic scattering and sky light, *In Proc. SIGGRAPH'96*, New Orleans, USA, 1996, pp. 379-386.

[4] N. Wang, Realistic and fast cloud rendering, *Journal of Graphics Tools*, Vol.9, No. 3, 2004, pp. 21-40.

[5] X. Y. Hu, B. Sun, X. H. Liang, An improved cloud rendering method, *In Proc. Image and GRAPHICS*, Xi'an, China, 2009, pp. 835-858.

[6] Y. Dobashi, K. Kaneda, T. Okita, T. Nishita, A simple, efficient method for realistic animation of clouds, *In Proc. Eurographics'00*, Brno, Czech Republic, 2000, pp. 19-28.

[7] R. Miyazaki, S. Yoshida, Y. Dobashi, A method for modeling clouds based on atmospheric fluid dynamics, *In Proc. Pacific Graphics'01*, Tokyo, Japan, 2001, pp. 363-372.

[8] M. J. Harris, W. V. Baxter, T. Scheuermann, Simulation of cloud dynamics on graphics hardware, *In Proc. Eurographics'03*, Sarajevo, Yugoslavia, 2003, pp. 92-101.

[9] M. J. Harris, Real-time cloud simulation and rendering, *PhD Dissertation*, University of North Carolina, 2003.

[10] J. Stam, Stable fluids, *In Proc. SIGGRAPH'99*, Los Angeles, USA, 1999, PP. 121-128.

[11] G. Y. Gardner, Visual simulation of cloud, *Computer Graphics*, Vol.19, No.3, 1985, pp. 279-303.

[12] M. J. Harris, A. Lastra, Real-time cloud rendering, *Computer Graphics Forum*, Vol.19, No.3, 2001, pp. 76-84.

[13] R. Miyazaki, Y. Dobashi, T. Nishita, A fast rendering method of clouds using shadow-view slices, *In Proc. CGIM'04*, Hawaii, USA, 2004, pp. 93-98..

[14] A. Bouthors, F. Neyret. S. Lefebvre, Real-time realistic illumination and shading of stratiform clouds, *In Proc. Eurographics Workshop on Natural Phenomena*, Vienna, Austria, 2006, pp. 1-10.

[15] A. Bouthors, F. Neyret, N. Max, E. Bruneton, C. Crassin, Interactive multiple anisotropic scattering in clouds, *In Proc. I3D'08*, Redwood City, USA, 2008, pp. 173-182.

[16] O. Elek, T. Ritschel, A. Wilkie, H. P. Seidel, Interactive cloud rendering using temporally-coherent photon mapping, *In Proc. Graphics Interface*, Toronto, Canada, 2012, pp. 141-148.

[17] K. Riley, D. Ebert, M. Kraus, Efficient rendering of atmospheric phenomena, *In Proc. Eurographics Symposium on Rendering*, Norkoping, Sweden, 2004, pp. 375-386.

[18] R. Miyazaki, Y. Dobashi, T. Nishita, Simulation of cumuliform clouds based on computational fluid dynamics, *In Proc. Eurographics'02*, Saarbruecken, Germany, 2002, pp. 405-410.

[19] Y. Dobashi, T. Yamamoto, T. Nishita, Interactive rendering of atmospheric scattering effects using graphics hardware, *In Proc. ACM Siggraph/Eurographics on Graphics Hardware*, Saarbruecken, Germany, 2002, pp. 99-107.

[20] R. Green, Spherical harmonics lighting: the gritty details, *In Proc. Game Developers*, San Jose, USA, 2003, pp. 1-47.

[21] T. Malzbender, Fourier volume rendering, *ACM Transactions on Graphics*, Vol.12, No.3, 1993, pp. 233-250.

[22] T. Totsuka, M. Levoy, Frequency domain volume rendering, *In Proc. SIGGRAPH'93*, Anaheim, USA, 1993, pp. 271-278.